ALIGNMENT-FREE SEQUENCE COMPARISON OVER HADOOP FOR COMPUTATIONAL BIOLOGY

Giuseppe Cattaneo, Gianluca Roscigno, **Umberto Ferraro Petrillo**, Raffaele Giancarlo

Sequence Comparison

- Given two genomic sequences

  $X = x_1, x_2, \ldots, x_n$
  $Y = y_1, y_2, \ldots, y_m$

  where x_i and y_i belong to an alphabet of symbols like {A,C,G,T}

- Determine how much similar X and Y are
- Identify regions of *similarity* between X and Y

Sequence Comparison Methods

- **Alignment-based Methods**
- **Alignment-free Methods**

Sequence Alignment Methods

- Try different arrengements for two or more sequences, so to identify regions of similarity
- Return a similarity score, stating how similar two sequences, or parts of them, are
- Example: local sequence alignment with scoring

... A G C T A G G T C C ...

... G A G C T A G G T C ...

... A G C T A G G T C C ...

... G A G C T A G G T C ...

... A G C T A G G T C C ...

... A G C T A G G T C T ...

- Well-studied, also from the experimental viewpoint
- Inefficient in terms of computational time

Alignment-free methods

- Extract a set of features from input sequences
- Similarity evaluated according to a distance function
- Example: sequence alignment with k-mers counting

... A B R A C A D A B R A ...

... R A C A D R A B R A B ...

... B E I J I N G ...

... A B R A C A D A B R A ...

... R A C A D R A B R A B ...

... A B R A C A D A B R A ...

... B E I J I N G ...

- Less accurate than alignment-based methods
- More efficient in terms of computational time

Objective of the Work

- **The problem:** Comparing big genomic sequences in a sequential setting may be very time-consuming, even for aligment-free methods

- **Our goal:**
  - Understand the performance issues of alignment-free methods in a sequential setting
  - Develop efficient and scalable alignment-free distributed methods (using MapReduce)

Outline of the talk

- **Part 1: Alignment-free Methods**
- **Part 2: The Sequential Approach**
- **Part 3: The Distributed approach**
- **Final remarks**

PART 1: ALIGNMENT-FREE METHODS

Alignment-free Methods based on K-mers Counts

- Let **X** be a sequence of characters
  - **k-mers of X:** all the substrings of length k existing in X
  - **k-mers frequency vector (i.e., K-mers count) for X:** the list of k-mers of X with associated frequencies
- Alignment-free methods evaluate the similarity between two sequences by comparing their k-mers frequency vector according to a **distance measure**

Step I: Extracting Frequency Vectors

A G C T A G G T C C ...

Given X and k:

```
for each k-mer in X
        if Freq[k-mer] is null
                Freq[k-mer] = 1
        else
                Freq[k-mer]++
```

Freq

C T A	1
A G C	1
G C T	1

Step II: Evaluating distance between Frequency Vectors

- Methods based on exact k-mers counts
  - E.g.: Squared Euclidean, D_2 Score, Feature Frequency Profile
- Methods based on approximate k-mers counts
  - E.g.: Spaced-Word Frequencies, Multiple Pattern Spaced-Words, Co-Phylog
- Euclidean Squared Function

$$d_{SE}(S,Q) = \sum_{i=1}^{n^k} (s_i - q_i)^2$$

PART 2: THE SEQUENTIAL APPROACH

A Software Framework for Alignment-free Algorithms

- Simplifies the development and the experimentation of alignment-free methods
- Operates in two steps
  - **Step 1:** Features set extraction
  - **Step 2:** Distance evaluation
- The only required code is about:
  - How features are represented
  - How features can be extracted from a sequence
  - How to evaluate the dissimilarity between features belonging to two distinct sequences
- Built-in support for a set of standard features and dissimilarity measurements (Squared Euclidean, D_2 Score, Feature Frequency Profile, Spaced-Word Frequencies, Multiple Pattern Spaced-Words, Co-Phylog)

Preliminary experiments

- **Experimental evaluation of euclidean squared distance**
  - **Sequences generated uniformly at random of increasing length (**≈50.000.000, ≈500.000.000, ≈1.500.000.000)
  - **Variable number of sequences** (5,10,15,20)
  - **Increasing values of k** (1,…,31)

- **Reference hardware:** AMD Opteron 2.2 Ghz with 4 Gb RAM

- **Outcomes:**
  - Execution time dominated by the extraction of frequency vectors → **Scalability Challenge**
  - Unable to test for k > 10 due to the huge memory usage of frequency vectors → **Feasibility Challenge**

PART 3: THE DISTRIBUTED APPROACH

The MapReduce paradigm

- A computing paradigm for data-intensive applications
- Useful when crunching big data sets through aggregation
- Computation takes place through two functions:
  - **map (in_key, in_value) -> list(out_key, intermediate_value)**
  - **reduce (out_key, list(intermediate_value)) -> list (out_key, out_value)**

K-mers alignment-free *via* MapReduce

- Computation split in two steps

- **Step 1: Frequency Vectors Extraction**
  - Map(idSeq, S) → list (kmer, (idSeq, 1))
  - Reduce(kmer, list(idSeq, 1)) →list (kmer, (idSeq, freq))

- **Step 2: Distance Evaluation**
  - Map(kmer, list(idSeq, freq)) → (idSeqA,idSeqB), (partDist, 1)
  - Reduce(idSeqA, idSeqB, list(partDist, 1)) → ((idSeqA,idSeqB), dist)

Optimizations

- **Optimization 1: Sequences I/O**
  - Input of sequences is managed by a custom file reader (SplitReader)
    - Small sequence files are aggregated into fewer and bigger files
    - Long sequences are virtually split in smaller chunks, each marked with a same id and processed by a separate map task

- **Optimization 2: In-memory Combiner**
  - K-mers found by map tasks are not immediately reported but buffered using a local temporary hash table

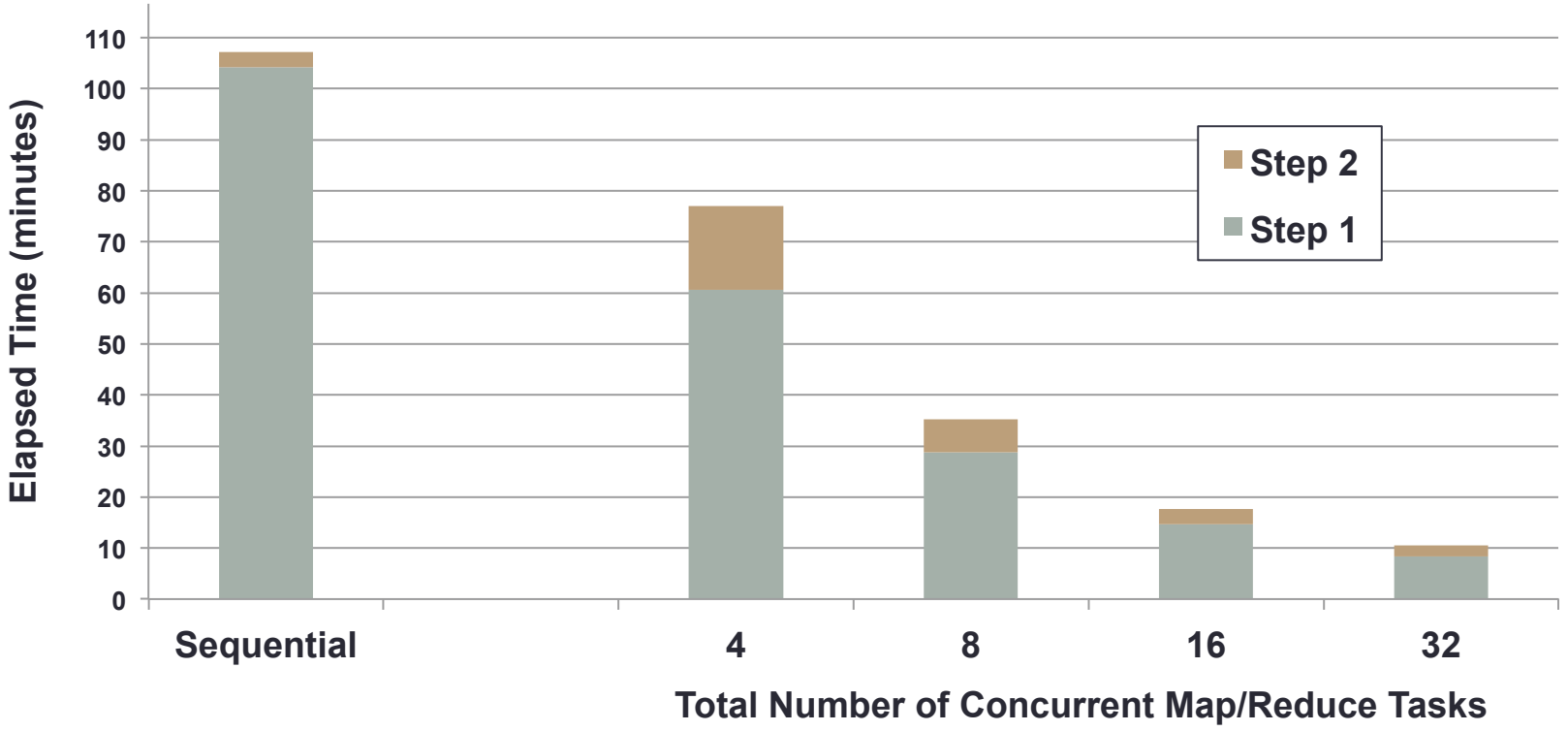
Distributed Experimental Settings

- Same sequential experiments repeated on Hadoop

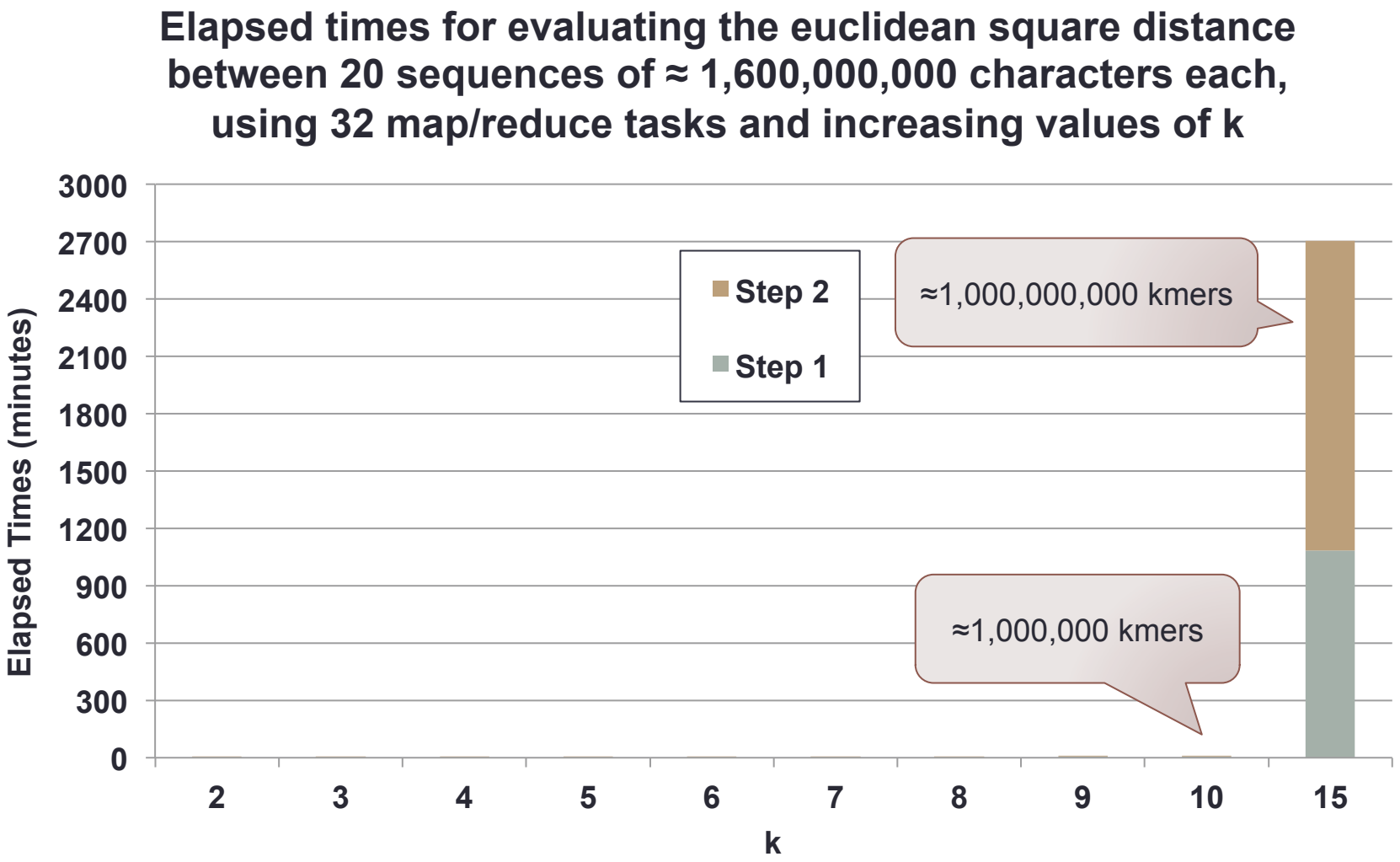
- **Reference hardware:** cluster of 8 AMD Opteron 2.2 Ghz PCs equipped with 32 cores and 128 Gigabyte of RAM, and connected by an Infiniband network
  - Up to total 32 concurrent map/reduce tasks (up to 4 per node)
  - HDFS replication factor set to 2
  - HDFS block size set to 128 Megabytes

Scalability Challenge

Elapsed Times for evaluating the euclidean square distance between 20 different sequences of ≈ 1,600,000,000 characters each, with k=10 and an increasing number of concurrent map/reduce tasks

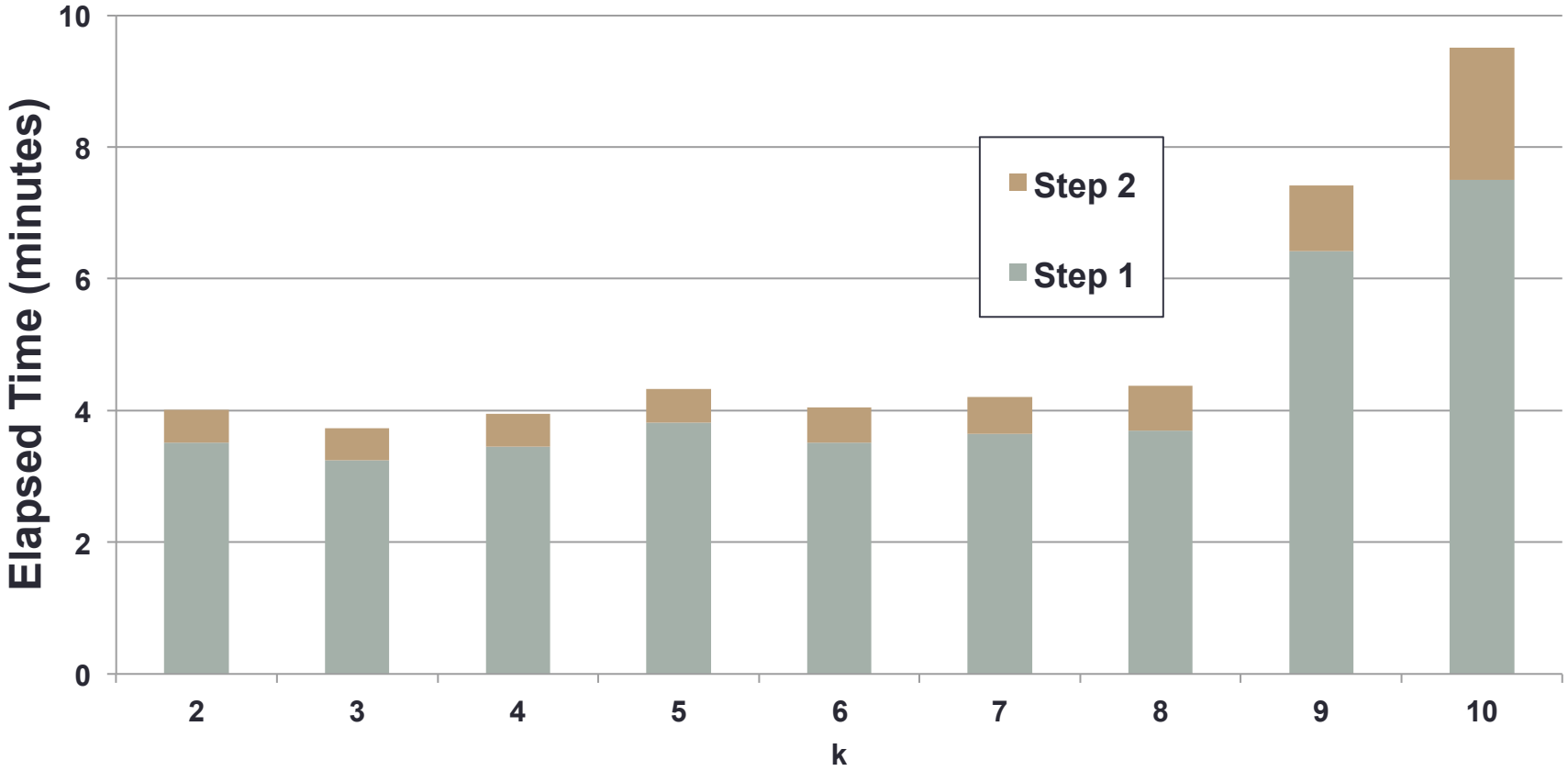


Feasability Challenge



Feasability Challenge

Elapsed times for evaluating the euclidean square distance between 20 sequences of ≈1,600,000,000 characters each, using 32 map/reduce tasks and increasing values of k



Final Remarks

- Alignment-free methods suffer from severe performance issues when run on very long sequences in a sequential setting
- Switching to MapReduce/Hadoop yelds scalable performance and helps in dealing with very long sequences, when using small values of k (≤10)
- Efficient processing of alignment-free methods with large values of k still an open problem. Possible optimizations:
  - **Implementation level:** Distributed Cache?
  - **Data distribution pattern level:** Reformulation of the MR step 2?
  - **Paradigm/Framework level:** Apache Spark?